Database Systems Project

Design Report

Online Language Learning Platform

08.04.2022

Project Group No: 33

Javid Moradi - 21903645

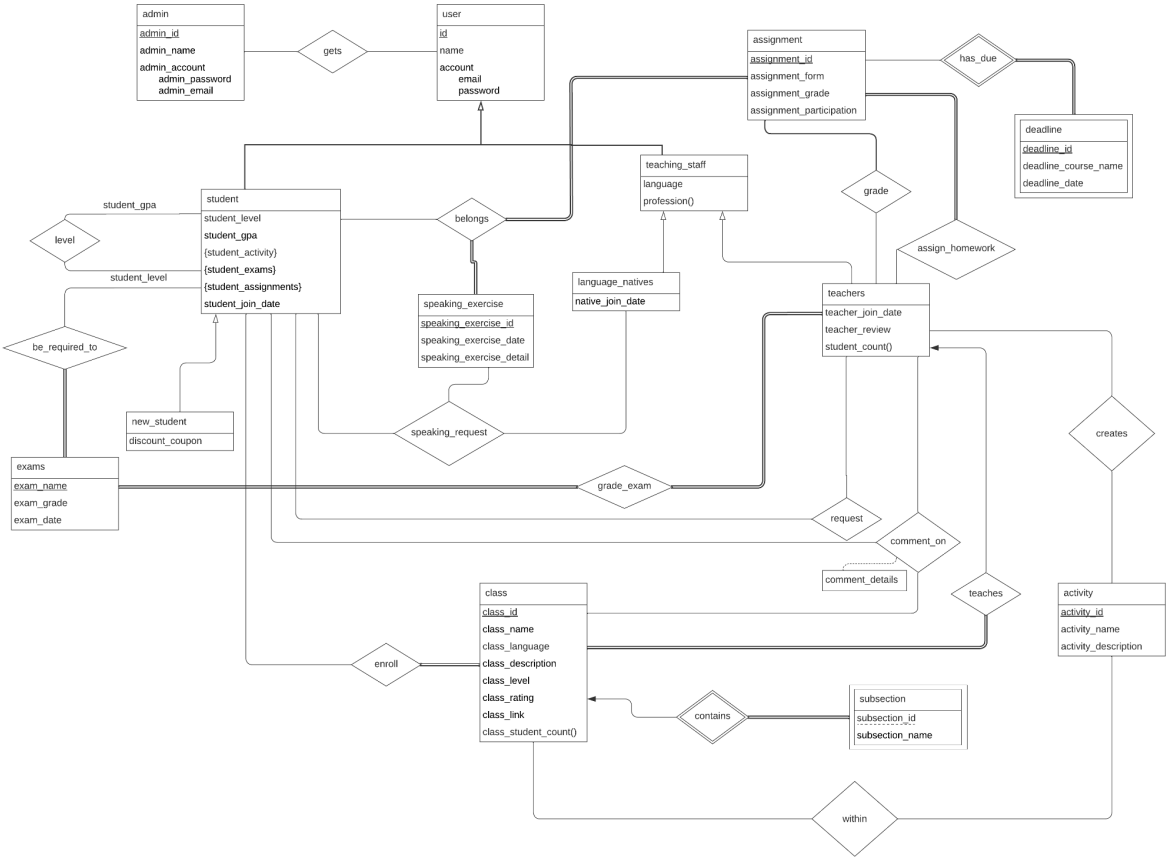
Mustafa Yuşa Bababdemez - 21703083

Nasuh Dinçer - 21702933

Tarık Buğra Karali -21703937

Course Instructor: Dr. Hamdi Dibeklioğlu - Course TA: Zülal Bingöl

1. Revised ER Model



For a better display of the Revised Entity-Relation Diagram, please refer to: https://imgur.com/a/dEqfDw7. Please note that an active internet connection is required for the given link. It is strongly recommended to download the image from the given link for the best quality of the diagram.

Link for this report:
https://nasuhdincer.github.io/Online-Language-Learning-Platform/

2. Relation Schemas

2.1 User

Model:

User(id, name, email, password)

Candidate Keys:

{(id), (email)}

Primary Key:

(id)

Functional Dependencies:

id → name,email,password

email → id,name,password

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE User(
id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(20) NOT NULL,
email VARCHAR(128) NOT NULL UNIQUE,
password VARCHAR(21) NOT NULL ) ENGINE = InnoDB;
```

2.2 Admin

Model:

User(admin_id, admin_name, admin_email, admin_password)

Foreign Key: admin_id references User(id)

Candidate Keys:

{(admin_id), (admin_email)}

Primary Key:

(admin_id)

Functional Dependencies:

admin_id → admin_name,admin_email,admin_password

admin_email → admin_id,admin_name,admin_password

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE Admin(
admin_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
admin_name VARCHAR(20) NOT NULL,
admin_email VARCHAR(128) NOT NULL UNIQUE,
admin_password VARCHAR(21) NOT NULL,
FOREIGN KEY (admin_id) REFERENCES User(id)) ENGINE = InnoDB;
```

2.3 Student

Model:

Student(student_id, student_level, student_gpa, student_join_date)

Foreign Key: student_id references User(id)

Student_activity(id, activity_id)

Student_exams(id, student_exams)

Student_assignments(id, assignment_id)

Candidate Keys:

{(student_id)}

Primary Key:

(student_id)

Functional Dependencies:

student_id→ student_level, student_gpa, student_join_date

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE Student(
student_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
student_level VARCHAR(20) NOT NULL,
student_gpa NUMERIC(3,2) NOT NULL,,
student_join_date DATE NOT NULL,
FOREIGN KEY (student_id) REFERENCES User(id)) ENGINE = InnoDB;
```

```
CREATE TABLE Student_activity(
student_id INT NOT NULL PRIMARY KEY,
activity_id INT NOT NULL PRIMARY KEY ) ENGINE = InnoDB;

CREATE TABLE Student_exams(
student_id INT NOT NULL PRIMARY KEY,
student_exams VARCHAR(20) NOT NULL PRIMARY KEY) ENGINE =
InnoDB;

CREATE TABLE Student_assignments(
student_id INT NOT NULL PRIMARY KEY,
assignment_id INT  NOT NULL PRIMARY KEY ) ENGINE = InnoDB;
```

2.4 Teaching Staff

Model:

teaching_staff(teaching_staff_id, language)

Foreign Key: teaching_staff_id references User(id)

Candidate Keys:

{(teaching_staff_id)}

Primary Key:

(teaching_staff_id)

Functional Dependencies:

teaching_staff_id→ language

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE teaching_staff(
teaching_staff_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
language VARCHAR(20) NOT NULL,
FOREIGN KEY (teaching_staff_id) REFERENCES User(id)) ENGINE =
InnoDB;
```

2.5 Language Natives

Model:

language_natives(language_natives_id, native_join_date)

Foreign Key: language_natives_id references User(id)

Candidate Keys:

{(language_natives_id)}

Primary Key:

(ilanguage_natives_id)

Functional Dependencies:

language_natives_id → native_join_date

Normal Form:

BCNF

Table Definition:

CREATE TABLE language_natives(

language_natives_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,

native_join_date DATE NOT NULL,

FOREIGN KEY (language_natives_id) REFERENCES User(id)) ENGINE = InnoDB;

2.6 Teachers

Model:

teachers(teacher_id, teacher_review, teacher_join_date)

Foreign Key: teacher_id references User(id)

Candidate Keys:

{(teacher_id)}

Primary Key:

(teacher_id)

Functional Dependencies:

teacher_id→ teacher_review

Normal Form:

BCNF

Table Definition:

CREATE TABLE teacher(

teacher_idINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
teacher_review VARCHAR(200) NOT NULL,
teacher_join_date DATE NOT NULL,
FOREIGN KEY (teacher_id) REFERENCES User(id)) ENGINE = InnoDB;

2.7 Assignment

Model:

assignment(assignment_id, assignment_form, assignment_grade, assignment_participation)

Candidate Keys:

{(assignment_id)}

Primary Key:

(assignment_id)

Functional Dependencies:

assignment_id→ assignment_form, assignment_grade, assignment_participation

Normal Form:

BCNF

Table Definition:

CREATE TABLE assignment(
id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
assignment_form VARCHAR(20) NOT NULL,
assignment_grade INT,
assignment_participation INT NOT NULL) ENGINE = InnoDB;

2.8 Speaking Exercise

Model:

speaking_exercise(speaking_exercise_id, speaking_exercise_date)

Candidate Keys:

{(speaking_exercise_id)}

Primary Key:

{(speaking_exercise_id)}

Functional Dependencies:

speaking_exercise_id → speaking_exercise_date

Normal Form:

BCNF

Table Definition:

CREATE TABLE speaking_exercise (
speaking_exercise_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
speaking_exercise_date DATE NOT NULL) ENGINE = InnoDB;

2.9 New Student

Model:

new_student(new_student_id, discount_coupon)
Foreign Key: new_student_id references User(id)

Candidate Keys:

{(new_student_id)}

Primary Key:

{(new_student_id)}

Functional Dependencies:

new_student_id→ discount_coupon

Normal Form:

BCNF

Table Definition:

CREATE TABLE speaking_exercise (
new_student_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
discount_coupon VARCHAR(20) NOT NULL,
FOREIGN KEY (new_student_id) REFERENCES User(id)) ENGINE =
InnoDB;

2.10 Deadline

Model:

deadline(deadline_id, deadline_course_name,deadline_date)

Candidate Keys:

{(deadline_id)}

Primary Key:

(deadline_id)

Functional Dependencies:

deadline_id→ deadline_course_name,deadline_date

Normal Form:

BCNF

Table Definition:

CREATE TABLE speaking_exercise (

deadline_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,

deadline_course_name VARCHAR(20) NOT NULL,

deadline_date DATE NOT NULL) ENGINE = InnoDB;

2.11 Class

Model:

class(class_id, class_name, class_language, class_description, class_level, class_rating, class_link)

Candidate Keys:

{(class_id)}

Primary Key:

(class_id)

Functional Dependencies:

class_id → class_name, class_language, class_description, class_level, class_rating, class_link

class_name→ class_id , class_language, class_description, class_level, class_rating, class_link

Normal Form:

BCNF

Table Definition:

CREATE TABLE class(

class_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,

class_name VARCHAR(20) NOT NULL UNIQUE,

class_language VARCHAR(20) NOT NULL,

class_description VARCHAR(200) NOT NULL,

class_level VARCHAR(20) NOT NULL,

class_rating NUMERIC(2,1) NOT NULL,

class_link VARCHAR(200) NOT NULL) ENGINE = InnoDB;

2.12 Subsection

Model:

subsection(subsection_id, subsection_name, class_id)

Foreign Key: class_id references class(class_id)

Candidate Keys:

{(subsection_id)}

Primary Key:

(subsection_id, class_id)

Functional Dependencies:

subsection_id→ subsection_name, class_id

Normal Form:

BCNF

Table Definition:

CREATE TABLE subsection(

subsection_id INT NOT NULL AUTO_INCREMENT,

subsection_name VARCHAR(20) NOT NULL,

class_id INT NOT NULL,

PRIMARY KEY (subsection_id,class_id),

FOREIGN KEY (class_id) REFERENCES class(class_id)) ENGINE =

InnoDB;

2.13 Exam

Model:

exam(exam_name, exam_grade, exam_date)

Candidate Keys:

{(exam_name)}

Primary Key:

(exam_name)

Functional Dependencies:

exam_name → exam_grade, exam_date

Normal Form:

BCNF

Table Definition:

CREATE TABLE exam(

exam_name VARCHAR(20) NOT NULL,

exam_grade VARCHAR(20) NOT NULL,

exam_date VARCHAR(20) NOT NULL,

PRIMARY KEY (exam_name)) ENGINE = InnoDB;

2.14 Activity

Model:

activity(activity_id, activity_name, activity_description)

Candidate Keys:

{(activity_id)}

Primary Key:

(activity_id)

Functional Dependencies:

activity_id → activity_name, activity_description

Normal Form:

BCNF

Table Definition:

CREATE TABLE activity(

activity_id INT NOT NULL,

activity_name VARCHAR(20) NOT NULL,

exam_description VARCHAR(20) NOT NULL,

PRIMARY KEY (activity_id)) ENGINE = InnoDB;

2.15 Has_due

Model:

has_due(assignment_id, deadline_id)

Foreign Key: assignment_id references assignment(assignment_id)

Foreign Key: deadline_id references deadline(deadline_id)

Candidate Keys:

None

Primary Key:

(assignment_id , deadline_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

```
CREATE TABLE has_due(
assignment_id INT NOT NULL,
deadline_id INT NOT NULL,
PRIMARY KEY (assignment_id , deadline_id ),
FOREIGN KEY (assignment_id  ) REFERENCES
assignment(assignment_id),
FOREIGN KEY (deadline_id) REFERENCES deadline(deadline_id)) ENGINE
= InnoDB;
```

2.16 Assign_homework

Model:

assign_homework(assignment_id, teacher_id)

Foreign Key: assignment_id references assignment(assignment_id)

Foreign Key: teacher_id references teachers(teacher_id)

Candidate Keys:

None

Primary Key:

(assignment_id , teacher_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

```
CREATE TABLE assign_homework(
assignment_id INT NOT NULL,
teacher_id INT NOT NULL,
PRIMARY KEY (assignment_id , teacher_id ),
FOREIGN KEY (assignment_id  ) REFERENCES
assignment(assignment_id),
FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id)) ENGINE =
InnoDB;
```

2.17 Grade

Model:

grade(assignment_id, teacher_id)

Foreign Key: assignment_id references assignment(assignment_id)

Foreign Key: teacher_id references teachers(teacher_id)

Candidate Keys:

None

Primary Key:

(assignment_id , teacher_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

```
CREATE TABLE grade(
assignment_id INT NOT NULL,
teacher_id INT NOT NULL,
PRIMARY KEY (assignment_id , teacher_id),
FOREIGN KEY (assignment_id  ) REFERENCES
assignment(assignment_id),
FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id)
) ENGINE = InnoDB;
```

2.18 Grade_exam

Model:

grade(exam_name, teacher_id)

Foreign Key: exam_name references exam(exam_name)

Foreign Key: teacher_id references teachers(teacher_id)

Candidate Keys:

None

Primary Key:

(exam_name , teacher_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

```
CREATE TABLE grade_exam(
exam_name VARCHAR(20) NOT NULL,
teacher_id INT NOT NULL,
PRIMARY KEY (exam_name, teacher_id ),
FOREIGN KEY (exam_name) REFERENCES exam(exam_name),
FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id)) ENGINE =
InnoDB;
```

2.19 Be_required_to

Model:

be_required_to(exam_name, student_id)

Foreign Key: exam_name references exam(exam_name)

Foreign Key: student_id references student(student_id)

Candidate Keys:

None

Primary Key:

(exam_name , student_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

```
CREATE TABLE be_required_to(
 exam_name VARCHAR(20) NOT NULL,
 student_id INT NOT NULL,
 PRIMARY KEY (exam_name, student_id ),
 FOREIGN KEY (exam_name) REFERENCES exam(exam_name),
 FOREIGN KEY (student_id) REFERENCES student(student_id)
) ENGINE = InnoDB;
```

2.20 Speaking_request

Model:

speaking_request(language_natives_id, student_id, speaking_exercise_id)

Foreign Key: language_natives_id references language_natives(language_natives_id)

Foreign Key: student_id references student(student_id)

Foreign Key: speaking_exercise_id references speaking_exercise(speaking_exercise_id)

Candidate Keys:

None

Primary Key:

(language_natives_id, student_id, speaking_exercise_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

```
CREATE TABLE speaking_request(
language_natives_id INT NOT NULL,
student_id INT NOT NULL,
speaking_exercise_id INT NOT NULL,
```

PRIMARY KEY (language_natives_id, student_id, speaking_exercise_id),
FOREIGN KEY (language_natives_id) REFERENCES language_natives
(language_natives_id),
FOREIGN KEY (student_id) REFERENCES student(student_id),
FOREIGN KEY (speaking_exercise_id) REFERENCES
speaking_exercise(speaking_exercise_id)) ENGINE = InnoDB;

2.21 Teaches

Model:

teaches(class_id, teacher_id)
Foreign Key: class_id references class(class_id)
Foreign Key: teacher_id references teachers(teacher_id)

Candidate Keys:

None

Primary Key:

(class_id , teacher_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

CREATE TABLE teaches(
class_id INT NOT NULL,
teacher_id INT NOT NULL,
PRIMARY KEY (class_id , teacher_id),
FOREIGN KEY (class_id) REFERENCES class(class_id),
FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id)
) ENGINE = InnoDB;

2.22 Within

Model:

within(class_id, activity_id)

Foreign Key: class_id references class(class_id)

Foreign Key: activity_id references activity(activity_id)

Candidate Keys:

None

Primary Key:

(class_id , activity_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

```
CREATE TABLE within(
class_id  INT NOT NULL,
activity_id INT NOT NULL,
PRIMARY KEY (class_id , activity_id),
FOREIGN KEY (class_id) REFERENCES class(class_id ),
FOREIGN KEY (activity_id ) REFERENCES activity(activity_id)
) ENGINE = InnoDB;
```

2.23 Enroll

Model:

within(class_id, student_id)

Foreign Key: class_id references class(class_id)

Foreign Key: student_id references student(student_id)

Candidate Keys:

None

Primary Key:

(class_id, student_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

```
CREATE TABLE enroll(
class_id  INT NOT NULL,
student_id INT NOT NULL,
PRIMARY KEY (class_id , student_id),
FOREIGN KEY (class_id) REFERENCES class(class_id ),
FOREIGN KEY (student_id) REFERENCES student(student_id)
) ENGINE = InnoDB;
```

2.24 Comment_on

Model:

comment_on(class_id, teacher_id, comment_details)

Foreign Key: class_id references class(class_id)

Foreign Key: teacher_id references teachers(teacher_id)

Candidate Keys:

None

Primary Key:

(class_id, teacher_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

```
CREATE TABLE comment_on(
class_id  INT NOT NULL,
teacher_id INT NOT NULL,
comment_details VARCHAR(200) NOT NULL,
PRIMARY KEY (class_id , teacher_id ),
 FOREIGN KEY (class_id) REFERENCES class(class_id ),
 FOREIGN KEY (teacher_id) REFERENCES teacher(teacher_id )
) ENGINE = InnoDB;
```

2.25 Belongs

Model:

belongs(student_id, speaking_exercise_id, assignment_id)

Foreign Key: student_id references student(student_id)

Foreign Key: speaking_exercise_id references speaking_exercise(speaking_exercise_id)

Foreign Key: assignment_id references assignment(assignment_id)

Candidate Keys:

None

Primary Key:

(student_id, speaking_exercise_id, assignment_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

```
CREATE TABLE belongs(
student_id INT NOT NULL,
speaking_exercise_id INT,
assignment_id INT,
PRIMARY KEY (student_id, speaking_exercise_id,assignment_id ),
FOREIGN KEY (assignment_id ) REFERENCES assignment(assignment_id),
FOREIGN KEY (student_id) REFERENCES student(student_id),
FOREIGN KEY (speaking_exercise_id ) REFERENCES
speaking_exercise(speaking_exercise_id )) ENGINE = InnoDB;
```

2.26 Request

Model :

request(student_id, teacher_id)

Foreign Key: student_id references student(student_id)

Foreign Key: teacher_id references teacher(teacher_id)

Candidate Keys:

None

Primary Key:

(student_id , teacher_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

CREATE TABLE request(
student_id INT NOT NULL,
teacher_id INT NOT NULL,
PRIMARY KEY (student_id , teacher_id),
FOREIGN KEY (student_id) REFERENCES student(student_id),
FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id)) ENGINE = InnoDB;

2.27 Creates

Model:

creates(assignment_id, teacher_id)
Foreign Key: assignment_id references assignment(assignment_id)
Foreign Key: teacher_id references teacher(teacher_id)

Candidate Keys:

None

Primary Key:

(assignment_id , teacher_id)

Functional Dependencies:

None

Normal Form:

None

Table Definition:

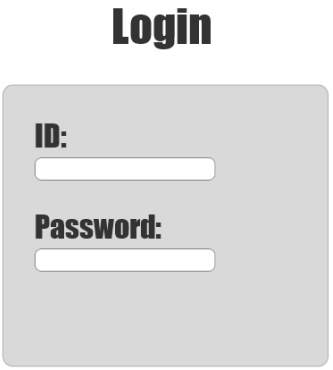
CREATE TABLE creates(
assignment_id INT NOT NULL,
teacher_id INT NOT NULL,

```
PRIMARY KEY (assignment_id, teacher_id),
FOREIGN KEY (assignment_id ) REFERENCES assignment(assignment_id),
FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id)) ENGINE =
InnoDB;
```

3. UI and Corresponding SQLs

Please note that this section only serves a mock-up of the final product, and there might be major differences, mostly in terms of interface, between this version and the final one.

3.1 Login Page



Please note that there will be no sign-up page for the project, as they will be created and be provided by the admins. After a user provides their credentials, the given credits will be compared to the set of users, and if a match is found, the user will simply login; otherwise, the user will not be redirected in.

SQL for Student

```
SELECT account_password
FROM student
WHERE account_password = {{ password }};
```

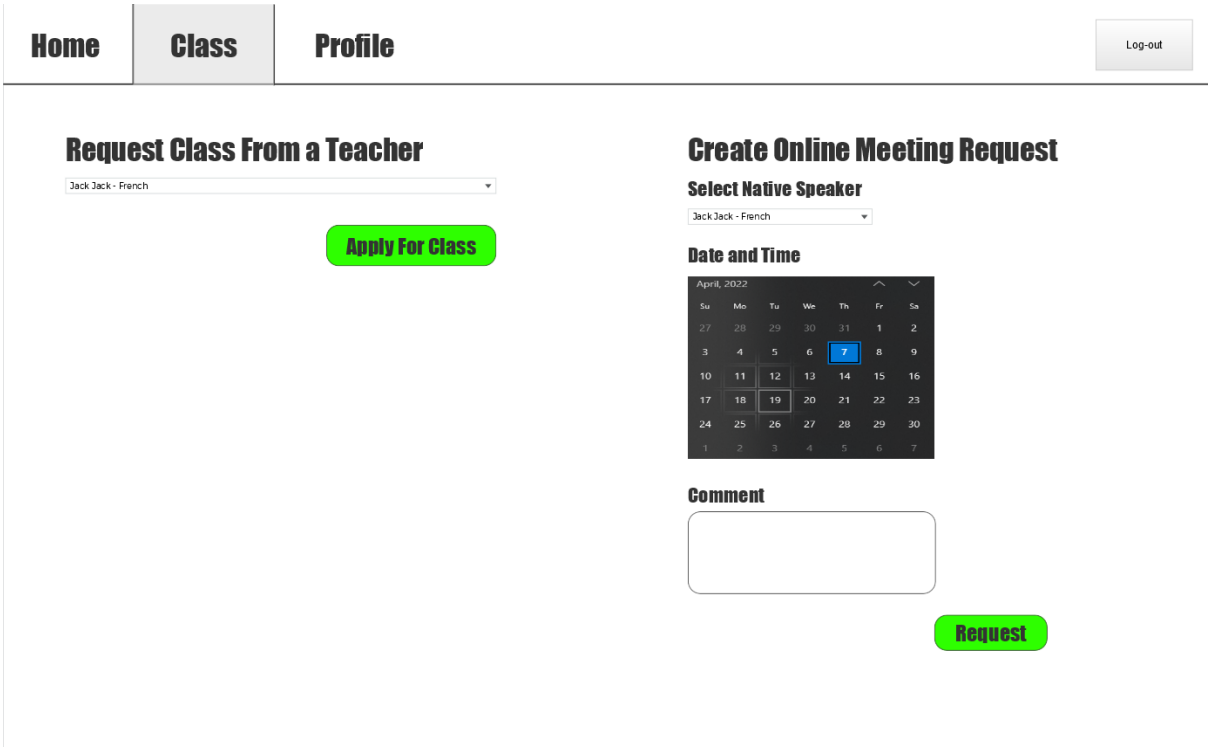
SQL for Teaching Staff

```
SELECT account_password
FROM teaching_staff
WHERE account_password = {{ password }};
```

SQL for Admin

```
SELECT admin_account_admin_password
FROM admin
WHERE admin_account_admin_password = {{ password }};
```

3.2 Student Class Page



On this page, a student can request a class from a teacher by specifying it from the dropdown bar. Also, meeting requests with a native speaker may be utilized by providing the required fields.

SQL for Requesting Class

```
SELECT class_name, name
FROM class NATURAL JOIN teaches NATURAL JOIN teachers
WHERE language = {{ language }} AND name = {{ name }};
```

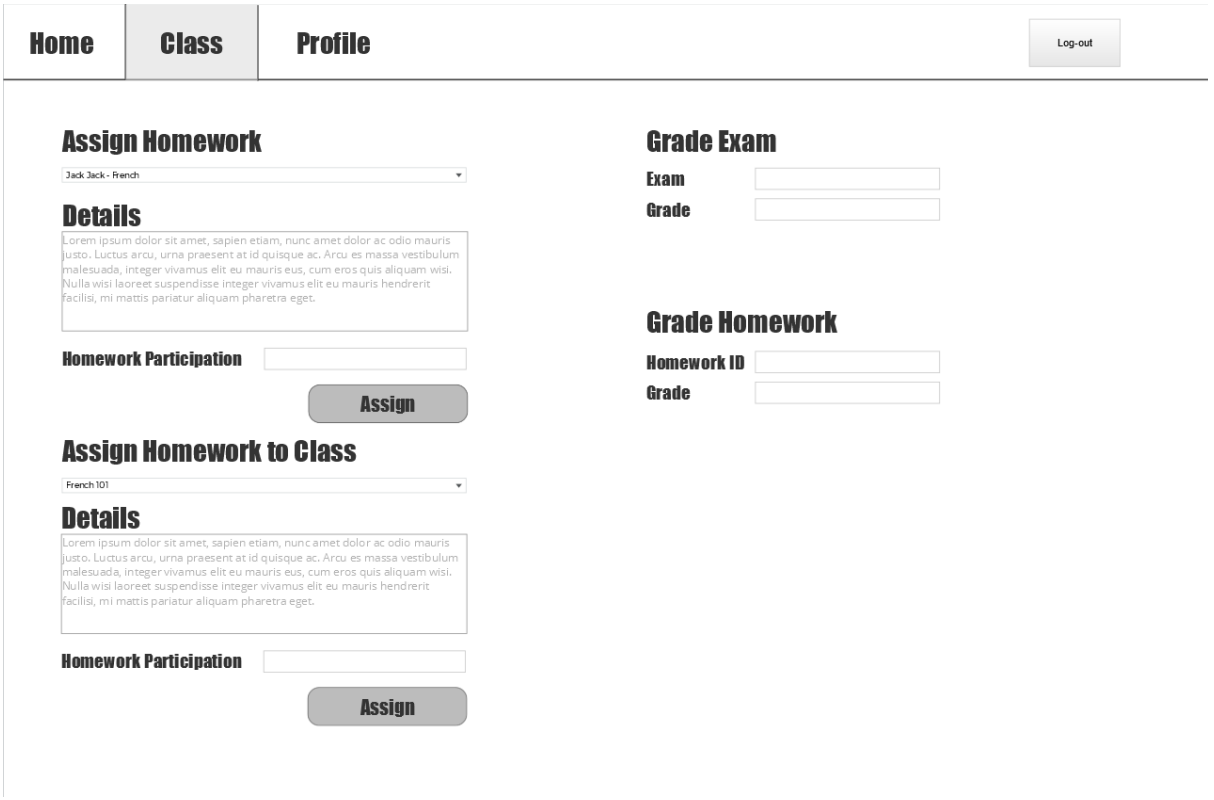
SQL for Online Meeting

```
SELECT language, name
```

```
FROM language_natives NATURAL JOIN speaking_request NATURAL JOIN
speaking_exercise
WHERE {{ date }} = speaking_exercise_date;
```

Note that the query for Online Meeting might be necessary to make sure that the selected language native does not hold any other exercise for the specified date.

3.3 Teacher Class Page



On this page, a teacher can assign homework(s) to the student(s), or grade the exam/assignment of a student. Note that the id of each exam and homework is unique. That means, each student has a unique correspondence with their homework/exam; thus, only specifying the id of the exam or assignment would be enough for the teacher.

SQL for Assigning Homework a Student

```
INSERT INTO assignment VALUES({{ assignment_id }}, {{ comment }},
null, {{ participation }});
INSERT INTO belongs VALUES({{ assignment_id }}, {{ id }});
```

SQL for Assigning Homework to a Class

```
INSERT INTO assignment VALUES({{ assignment_id }}, {{ comment }},
null, {{ participation }});
INSERT INTO assign_homework VALUES({{ user_id }}, {{ assignment_id
}});
```

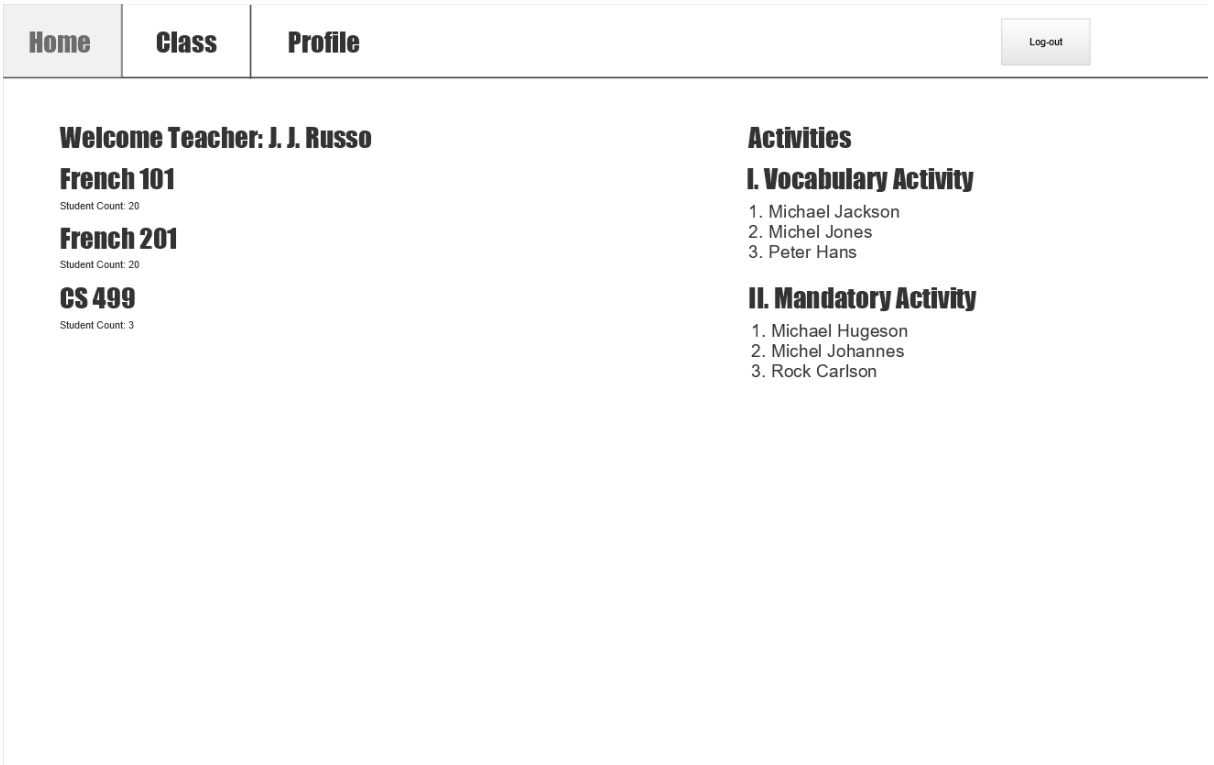
SQL for Grading an Exam

```
UPDATE exams
SET exam_grade = {{ grade }}
WHERE exam_name = {{ exam }};
```

SQL for Grading a Homework

```
UPDATE assignment
SET exam_grade = {{ grade }}
WHERE assignment_id = {{ homework_ID }};
```

3.4 Teacher Home Page



On the homepage of a teacher, the classes, with their student count, and activities of a teacher's students, are displayed. The classes are the offered courses of the teacher.

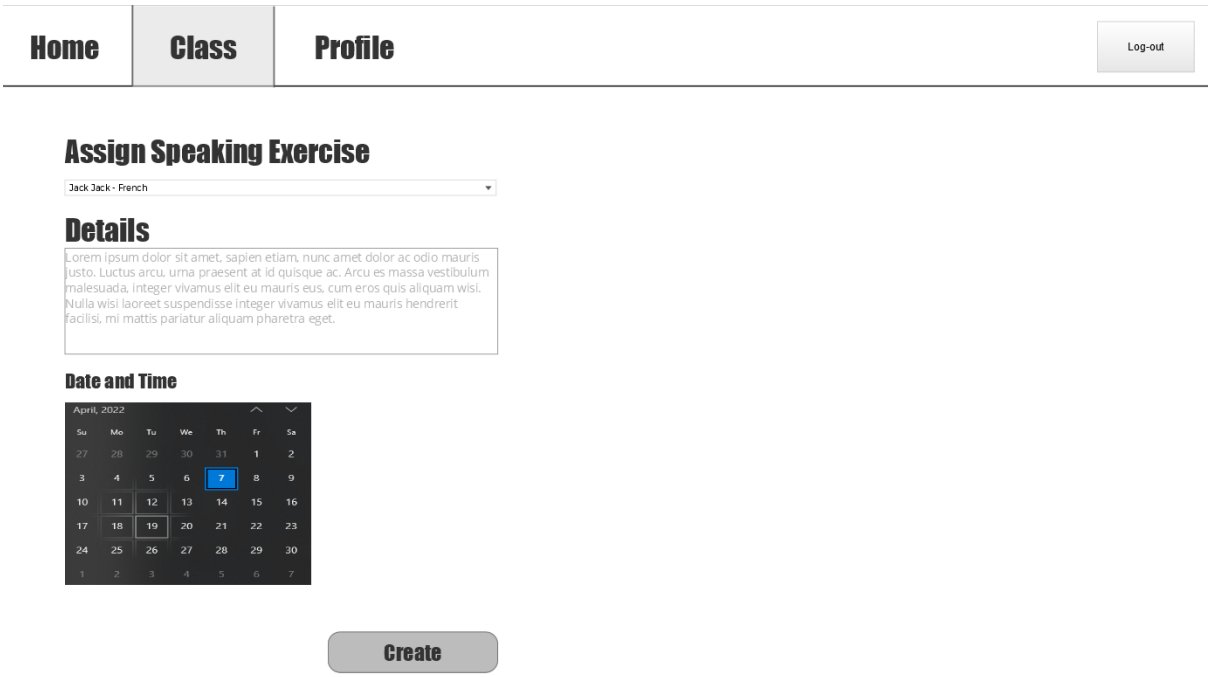
SQL for Displaying the Class and Student Count of a Teacher

```
SELECT class_name, count(id)
FROM class NATURAL JOIN enroll NATURAL JOIN student NATURAL JOIN
teaches NATURAL JOIN teachers S
GROUP BY class_id
WHERE S.id = {{ user_id }};
```

SQL for Displaying a Teacher's Activities

```
SELECT S.student_activity, S.name
FROM student S NATURAL JOIN request R NATURAL JOIN teachers T
WHERE T.id = {{ user_id }}
GROUP BY S.student_activity;
```

3.5 Language Native Class Page

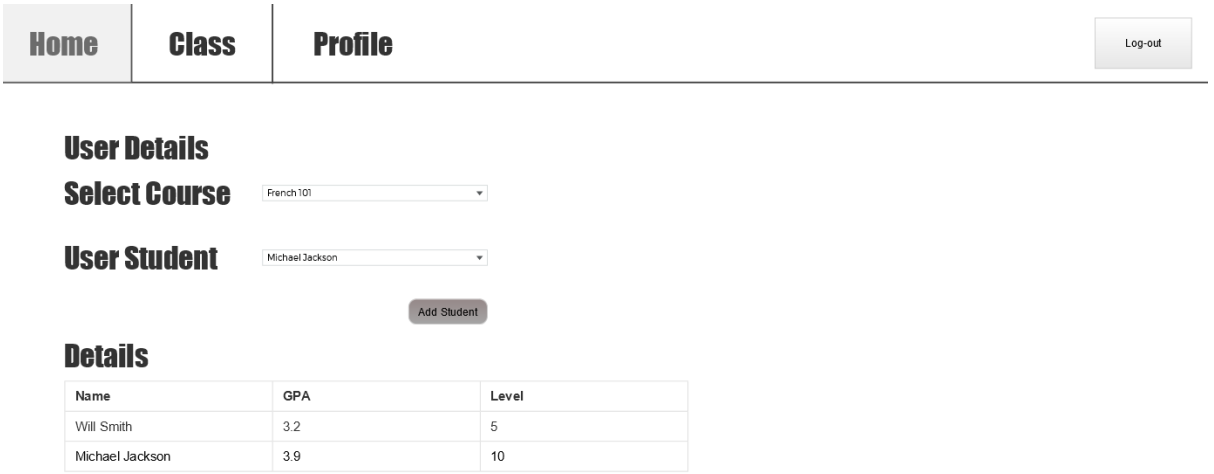


Language natives can create speaking exercises for students from their class pages by specifying required fields.

SQL for Assigning a Speaking Exercise to a Student

```
INSERT INTO speaking_exercise VALUES({{ speaking_exercise_id }}, {{
date }}, {{ comment }});
INSERT INTO speaking_request VALUES({{ user_id }}, {{
speaking_exercise_id }});
```

3.6 Admin Home Page



An admin can select a user from the system and look at the user's details for analysis purposes. Further usage/utilization of the ‘analysis’ has not been decided yet; thus, for simplicity and demonstration purposes, only the query for retrieval is given.

SQL to Get a Student

```
SELECT name, student_level, student_gpa
FROM student NATURAL JOIN enroll NATURAL JOIN class
WHERE name = {{ user_student }} AND class_name = {{ name }};
```